bdiGDB

JTAG debug interface for GNU Debugger

PowerPC 7440 / 7450 / 86xx

User Manual

Manual Version 1.02 for BDI3000

abatron

©1997-2015 by Abatron AG

1 Introduction 3
1.1 BDI3000 3
1.2 BDI Configuration 4

2 Installation 5
2.1 Connecting the BDI3000 to Target 5
2.2 Connecting the BDI3000 to Power Supply 7
2.3 Status LED «MODE» 8
2.4 Connecting the BDI3000 to Host 9
2.4.1 Serial line communication 9
2.4.2 Ethernet communication 10
2.5 Installation of the Configuration Software 11
2.5.1 Configuration with a Linux / Unix host 12
2.5.2 Configuration with a Windows host 14
2.5.3 Configuration via Telnet / TFTP 16
2.6 Testing the BDI3000 to host connection 18
2.7 TFTP server for Windows 18

3 Using bdiGDB 19
3.1 Principle of operation 19
3.2 Configuration File 21
3.2.1 Part [INIT] 22
3.2.2 Part [TARGET] 24
3.2.3 Part [HOST] 29
3.2.4 Part [FLASH] 31
3.2.5 Part [REGS] 35
3.3 Debugging with GDB 37
3.3.1 Target setup 37
3.3.2 Connecting to the target 37
3.3.3 Breakpoint Handling 38
3.3.4 GDB monitor command 38
3.3.5 Target serial I/O via BDI 39
3.3.6 Embedded Linux MMU Support 40
3.4 Telnet Interface 42
3.5 Dual-Core Support for MPC8641D 44

4 Specifications 46

5 Environmental notice 47

6 Declaration of Conformity (CE) 47

7 Warranty and Support Terms 48
7.1 Hardware 48
7.2 Software 48
7.3 Warranty and Disclaimer 48
7.4 Limitation of Liability 48

Appendices

A Troubleshooting 49

B Maintenance 50

C Trademarks 50

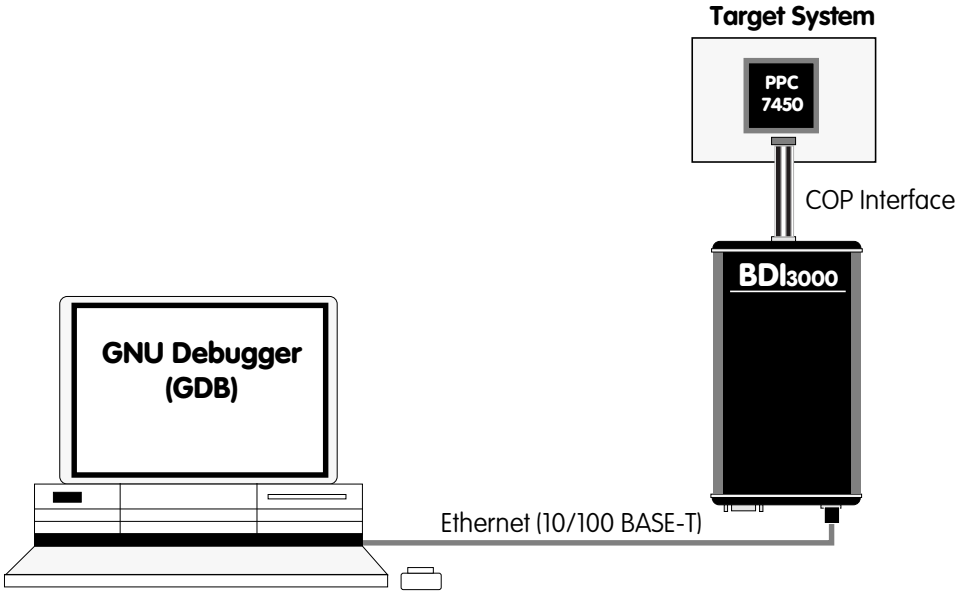
© Copyright 1997-2015 by ABATRON AG Switzerland

1 Introduction

bdiGDB enhances the GNU debugger (GDB), with JTAG/COP debugging for PowerPC 7440/7450/86xx based targets. With the built-in Ethernet interface you get a very fast code download speed. No target communication channel (e.g. serial line) is wasted for debugging purposes. Even better, you can use fast Ethernet debugging with target systems without network capability. The host to BDI communication uses the standard GDB remote protocol.

An additional Telnet interface is available for special debug tasks (e.g. force a hardware reset, program flash memory).

The following figure shows how the BDI3000 interface is connected between the host and the target:



1.1 BDI3000

The BDI3000 is the main part of the bdiGDB system. This small box implements the interface between the JTAG pins of the target CPU and a 10/100Base-T Ethernet connector. The firmware of the BDI3000 can be updated by the user with a simple Linux/Windows configuration program or interactively via Telnet/TFTP. The BDI3000 supports 1.2 – 5.0 Volts target systems.

© Copyright 1997-2015 by ABATRON AG Switzerland

1.2 BDI Configuration

As an initial setup, the IP address of the BDI3000, the IP address of the host with the configuration file and the name of the configuration file is stored within the flash of the BDI3000.
Every time the BDI3000 is powered on, it reads the configuration file via TFTP.

Following an example of a typical configuration file:

```
;bdiGDB configuration file for Sandpoint 7450 evaluation system
;-------------------------------------------------------------
;
[INIT]
; init core register
;WREG    MSR           0x00000000     ;clear MSR

; init memory controller (based on DINK32)
WM32    0xFEC00000      0x80000080      ;select MSAR1
WM32    0xFEE00000      0x00204060      ;
WM32    0xFEC00000      0x84000080      ;select MSAR2
WM32    0xFEE00000      0x80a0c0e0      ;
WM32    0xFEC00000      0x90000080      ;select MEAR1
WM32    0xFEE00000      0x1f3f5f7f      ;
WM32    0xFEC00000      0x94000080      ;select MEAR2
WM32    0xFEE00000      0x9fbfdfff      ;
WM32    0xFEC00000      0xa0000080      ;select MBEN
WM8     0xFEE00000      0x03            ;
WM32    0xFEC00000      0xa0000080      ;select MPM
WM8     0xFEE00003      0x32            ;
WM32    0xFEC00000      0xf0000080      ;select MCCR1
WM32    0xFEE00000      0x0000e075      ;do not set MEMGO
WM32    0xFEC00000      0xf4000080      ;select MCCR2
WM32    0xFEE00000      0xcc044004      ;
WM32    0xFEC00000      0xf8000080      ;select MCCR3
WM32    0xFEE00000      0x00004078      ;
WM32    0xFEC00000      0xfc000080      ;select MCCR4
WM32    0xFEE00000      0x39323235      ;
WM32    0xFEC00000      0xf0000080      ;select MCCR1
WM32    0xFEE00000      0x0000e875      ;now set MEMGO
;
WM32    0xFEC00000      0x78000080      ;select EUMBBAR
WM32    0xFEE00000      0x000000fc      ;Embedded utility memory block at 0xFC000000
;
WM32    0xFEC00000      0xa8000080      ;select PICR1
WM32    0xFEE00000      0x901014ff      ;enable flash write (Flash on processor bus)
;
[TARGET]
CPUTYPE     7450        ;the CPU type (7450)
JTAGCLOCK   1           ;use 16 MHz JTAG clock
WORKSPACE   0x00000000  ;workspace in target RAM for data cache flush and L3PM access
BDIMODE     AGENT       ;the BDI working mode (LOADONLY | AGENT | GATEWAY)
BREAKMODE   SOFT        ;SOFT or HARD, HARD uses PPC hardware breakpoint
DCACHE      NOFLUSH     ;data cache flushing (FLUSH | NOFLUSH)

[HOST]
IP                  151.120.25.115
FILE                E:\cygnus\root\usr\demo\mpc7450\vmlinux
FORMAT              IMAGE
LOAD                MANUAL          ;load code MANUAL or AUTO after reset
```

Based on the information in the configuration file, the target is automatically initialized after every reset.

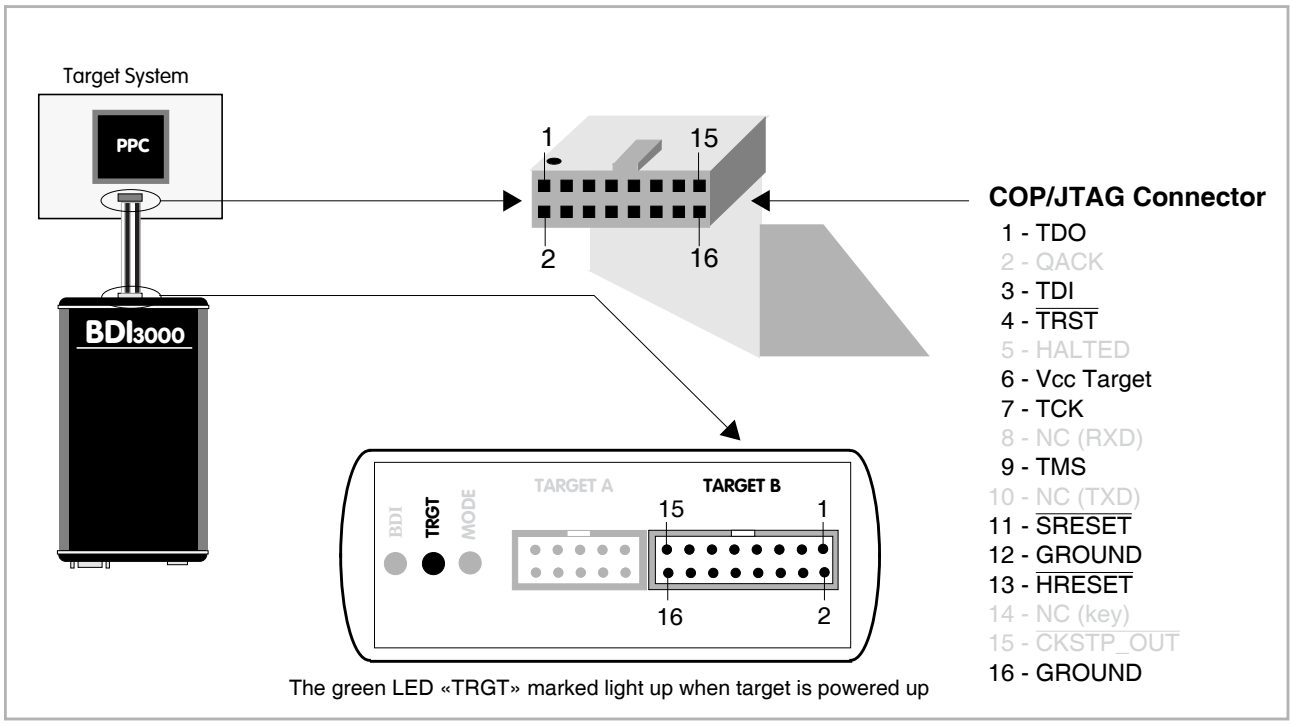
© Copyright 1997-2015 by ABATRON AG Switzerland

2 Installation

2.1 Connecting the BDI3000 to Target

The cable to the target system is a 16 pin flat ribbon cable. In case where the target system has an appropriate connector, the cable can be directly connected. The pin assignment is in accordance with the PowerPC COP connector specification.

In order to ensure reliable operation of the BDI (EMC, runtimes, etc.) the target cable length must not exceed 20 cm (8").



The green LED «TRGT» marked light up when target is powered up

For BDI TARGET B connector signals see table on next page.

Warning:
Before you can use the BDI3000 with an other target processor type (e.g. PPC <--> ARM), a new setup has to be done (see chapter 2.5). During this process the target cable must be disconnected from the target system.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming a new firmware for an other target CPU.

© Copyright 1997-2015 by ABATRON AG Switzerland

BDI TARGET B Connector Signals:

Pin	Name	Description
1	TDO	**JTAG Test Data Out** This input to the BDI3000 connects to the target TDO pin.
2	$\overline{\text{QACK}}$	**QACK** This output of the BDI3000 connects to the target QACK pin. By default this pin is not driven by the BDI3000. With an entry in the configuration file it can be forced low.
3	TDI	**JTAG Test Data In** This output of the BDI3000 connects to the target TDI pin.
4	$\overline{\text{TRST}}$	**JTAG Test Reset** This output of the BDI3000 resets the JTAG TAP controller on the target.
5	IN0	**General purpose Input** This input to the BDI3000 connects to the target HALTED pin. Currently not used.
6	Vcc Target	**1.2 – 5.0V:** This is the target reference voltage. It indicates that the target has power and it is also used to create the logic-level reference for the input comparators. It also controls the output logic levels to the target. It is normally fed from Vdd I/O on the target board.
7	TCK	**JTAG Test Clock** This output of the BDI3000 connects to the target TCK pin.
8	<reseved>	
9	TMS	**JTAG Test Mode Select** This output of the BDI3000 connects to the target TMS line.
10	<reseved>	
11	$\overline{\text{SRESET}}$	**Soft-Reset** This open collector output of the BDI3000 connects to the target SRESET pin.
12	GROUND	**System Ground**
13	$\overline{\text{HRESET}}$	**Hard-Reset** This open collector output of the BDI3000 connects to the target HRESET pin.
14	<reseved>	
15	IN1	**General purpose Input** This input to the BDI3000 connects to the target CKSTP_OUT pin. Currently not used.
16	GROUND	**System Ground**

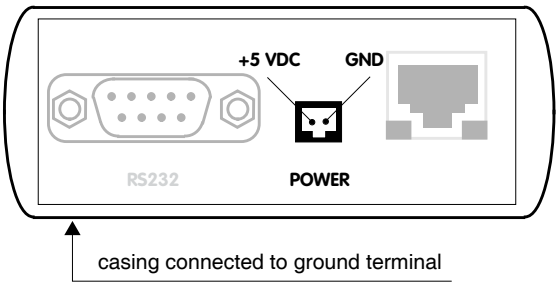
© Copyright 1997-2015 by ABATRON AG Switzerland

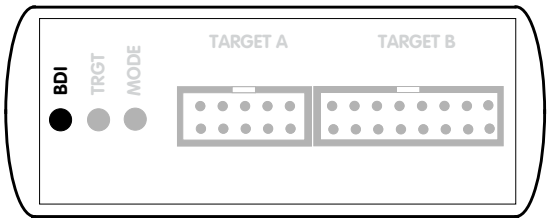
2.2 Connecting the BDI3000 to Power Supply

The BDI3000 needs to be supplied with the enclosed power supply from Abatron (5VDC).

Before use, check if the mains voltage is in accordance with the input voltage printed on power supply. Make sure that, while operating, the power supply is not covered up and not situated near a heater or in direct sun light. Dry location use only.

For error-free operation, the power supply to the BDI3000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**





The green LED «BDI» marked light up when 5V power is connected to the BDI3000

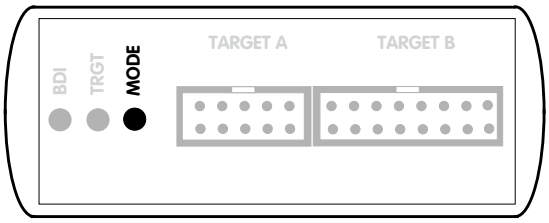
Please switch on the system in the following sequence:

- 1 –> external power supply
- 2 –> target system

© Copyright 1997-2015 by ABATRON AG Switzerland

2.3 Status LED «MODE»

The built in LED indicates the following BDI states:



MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The output voltage from the power supply is too low.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

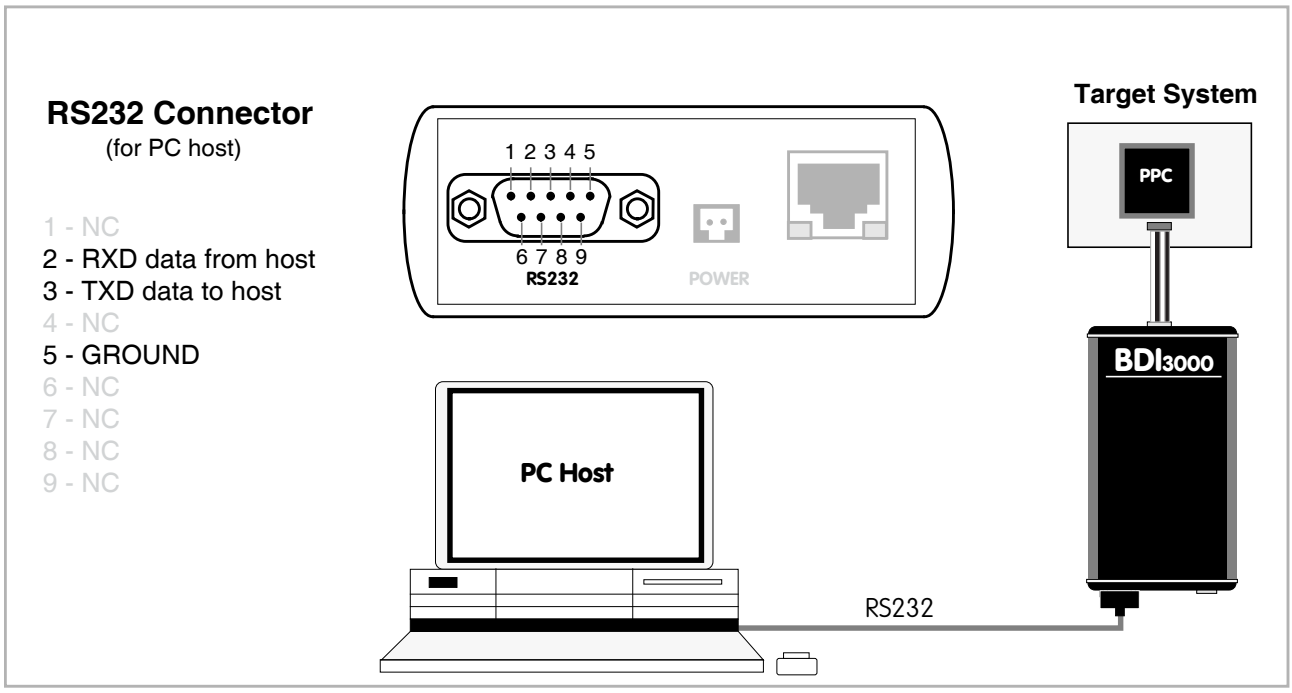
© Copyright 1997-2015 by ABATRON AG Switzerland

2.4 Connecting the BDI3000 to Host

2.4.1 Serial line communication

Serial line communication is only used for the initial configuration of the bdiGDB system.

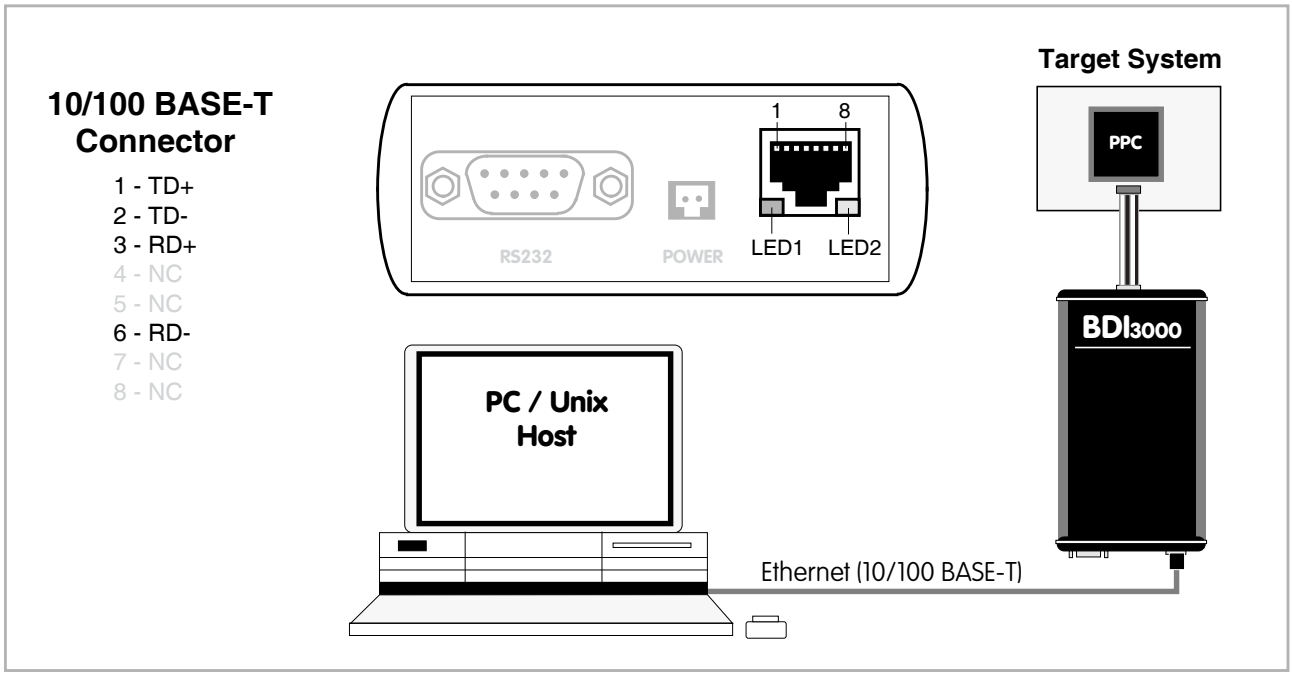
The host is connected to the BDI through the serial interface (COM1...COM4). The communication cable (included) between BDI and Host is a serial cable. There is the same connector pinout for the BDI and for the Host side (Refer to Figure below).



© Copyright 1997-2015 by ABATRON AG Switzerland

2.4.2 Ethernet communication

The BDI3000 has a built-in 10/100 BASE-T Ethernet interface (see figure below). Connect an UTP (Unshielded Twisted Pair) cable to the BD3000. Contact your network administrator if you have questions about the network.



The following explains the meanings of the built-in LED lights:

LED	Function	Description
LED 1 (green)	Link / Activity	When this LED light is ON, data link is successful between the UTP port of the BDI3000 and the hub to which it is connected. The LED blinks when the BDI3000 is receiving or transmitting data.
LED 2 (amber)	Speed	When this LED light is ON, 100Mb/s mode is selected (default). When this LED light is OFF, 10Mb/s mode is selected

© Copyright 1997-2015 by ABATRON AG Switzerland

2.5 Installation of the Configuration Software

On the enclosed diskette you will find the BDI configuration software and the firmware required for the BDI3000. For Windows users there is also a TFTP server included.

The following files are on the diskette.

b30pwsgd.exe	Windows Configuration program
b30pwsgd.xxx	Firmware for the BDI3000
tftpsrv.exe	TFTP server for Windows (WIN32 console application)
*.cfg	Configuration files
*.def	Register definition files
bdisetup.zip	ZIP Archive with the Setup Tool sources for Linux / UNIX hosts.

Overview of an installation / configuration process:

- Create a new directory on your hard disk
- Copy the entire contents of the enclosed diskette into this directory
- Linux only: extract the setup tool sources and build the setup tool
- Use the setup tool or Telnet (default IP) to load/update the BDI firmware
  Note: A new BDI has no firmware loaded.
- Use the setup tool or Telnet (default IP) to load the initial configuration parameters
  - IP address of the BDI.
  - IP address of the host with the configuration file.
  - Name of the configuration file. This file is accessed via TFTP.
  - Optional network parameters (subnet mask, default gateway).

Activating BOOTP:
The BDI can get the network configuration and the name of the configuration file also via BOOTP. For this simple enter 0.0.0.0 as the BDI's IP address (see following chapters). If present, the subnet mask and the default gateway (router) is taken from the BOOTP vendor-specific field as defined in RFC 1533.

With the Linux setup tool, simply use the default parameters for the -c option:

```
[root@LINUX_1 bdisetup]# ./bdisetup -c -p/dev/ttyS0 -b57
```

The MAC address is derived from the serial number as follows:
MAC: 00-0C-01-xx-xx-xx , replace the xx-xx-xx with the 6 left digits of the serial number
Example: SN# 33123407 ==>> 00-0C-01-33-12-34

Default IP: 192.168.53.72
Before the BDI is configured the first time, it has a default IP of 192.168.53.72 that allows an initial configuration via Ethernet (Telnet or Setup Tools). If your host is not able to connect to this default IP, then the initial configuration has to be done via the serial connection.

© Copyright 1997-2015 by ABATRON AG Switzerland

2.5.1 Configuration with a Linux / Unix host

The firmware update and the initial configuration of the BDI3000 is done with a command line utility. In the ZIP Archive bdisetup.zip are all sources to build this utility. More information about this utility can be found at the top in the bdisetup.c source file. There is also a make file included.
Starting the tool without any parameter displays information about the syntax and parameters.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.

Following the steps to bring-up a new BDI3000:

1. Build the setup tool:
The setup tool is delivered only as source files. This allows to build the tool on any Linux / Unix host. To build the tool, simply start the make utility.

```
[root@LINUX_1 bdisetup]# make
cc -O2   -c -o bdisetup.o bdisetup.c
cc -O2   -c -o bdicnf.o bdicnf.c
cc -O2   -c -o bdidll.o bdidll.c
cc -s bdisetup.o bdicnf.o bdidll.o -o bdisetup
```

2. Check the serial connection to the BDI:
With "bdisetup -v" you may check the serial connection to the BDI. The BDI will respond with information about the current loaded firmware and network configuration.
Note: Login as root, otherwise you probably have no access to the serial port.

```
$ ./bdisetup -v -p/dev/ttyS0 -b115
BDI Type : BDI3000 (SN: 30000154)
Loader   : V1.00
Firmware : unknown
MAC      : ff-ff-ff-ff-ff-ff
IP Addr  : 255.255.255.255
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 255.255.255.255
Config   : ÿÿÿÿÿÿÿ.....
```

3. Load/Update the BDI firmware:
With "bdisetup -u" the firmware is programmed into the BDI3000 flash memory. This configures the BDI for the target you are using. Based on the parameters -a and -t, the tool selects the correct firmware file. If the firmware file is in the same directory as the setup tool, there is no need to enter a -d parameter.

```
$ ./bdisetup -u -p/dev/ttyS0 -b115 -aGDB -tMPC7450
Connecting to BDI loader
Programming firmware with ./b30pwsgd.100
Erasing firmware flash ....
Erasing firmware flash passed
Programming firmware flash ....
Programming firmware flash passed
```

© Copyright 1997-2015 by ABATRON AG Switzerland

4. Transmit the initial configuration parameters:
With "bdisetup -c" the configuration parameters are written to the flash memory within the BDI.
The following parameters are used to configure the BDI:

BDI IP Address	The IP address for the BDI3000. Ask your network administrator for assigning an IP address to this BDI3000. Every BDI3000 in your network needs a different IP address.
Subnet Mask	The subnet mask of the network where the BDI is connected to. A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask. If the BDI and the host are in the same subnet, it is not necessary to enter a subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI3000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This file is read via TFTP. Keep in mind that TFTP has it's own root directory (usual /tftpboot). You can simply copy the configuration file to this directory and the use the file name without any path. For more information about TFTP use "man tftpd".

```
$ ./bdisetup -c -p/dev/ttyS0 -b115 \
> -i151.120.25.102 \
> -h151.120.25.112 \
> -fe:/bdi3000/mytarget.cfg
Connecting to BDI loader
Writing network configuration
Configuration passed
```

5. Check configuration and exit loader mode:
The BDI is in loader mode when there is no valid firmware loaded or you connect to it with the setup tool. While in loader mode, the Mode LED is blinking. The BDI will not respond to network requests while in loader mode. To exit loader mode, the "bdisetup -v -s" can be used. You may also power-off the BDI, wait some time (1min.) and power-on it again to exit loader mode.

```
$ ./bdisetup -v -p/dev/ttyS0 -b115 -s
BDI Type : BDI3000 (SN: 30000154)
Loader   : V1.00
Firmware : V1.00 bdiGDB for MPC7450
MAC      : 00-0c-01-30-00-01
IP Addr  : 151.120.25.102
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 151.120.25.112
Config   : /bdi3000/mytarget.cfg
```

The Mode LED should go off, and you can try to connect to the BDI via Telnet.

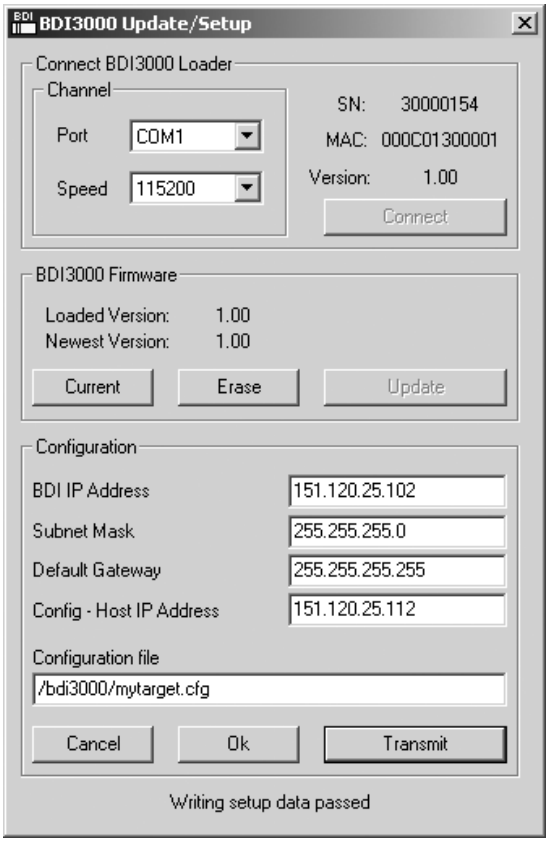
```
$ telnet 151.120.25.102
```

© Copyright 1997-2015 by ABATRON AG Switzerland

2.5.2 Configuration with a Windows host

First make sure that the BDI is properly connected (see Chapter 2.1 to 2.4).

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.



dialog box «BDI3000 Update/Setup»

Before you can use the BDI3000 together with the GNU debugger, you must store the initial configuration parameters in the BDI3000 flash memory. The following options allow you to do this:

Port	Select the communication port where the BDI3000 is connected during this setup session. If you select Network, make sure the Loader is already active (Mode LED blinking). If there is already a firmware loaded and running, use the Telnet command "boot loader" to activate Loader Mode.
Speed	Select the baudrate used to communicate with the BDI3000 loader during this setup session.
Connect	Click on this button to establish a connection with the BDI3000 loader. Once connected, the BDI3000 remains in loader mode until it is restarted or this dialog box is closed.
Current	Press this button to read back the current loaded BDI3000 firmware version. The current firmware version will be displayed.

© Copyright 1997-2015 by ABATRON AG Switzerland

Erase	Press this button to erase the current loaded firmware.
Update	This button is only active if there is a newer firmware version present in the execution directory of the bdiGDB setup software. Press this button to write the new firmware into the BDI3000 flash memory.
BDI IP Address	Enter the IP address for the BDI3000. Use the following format: xxx.xxx.xxx.xxx e.g.151.120.25.101 Ask your network administrator for assigning an IP address to this BDI3000. Every BDI3000 in your network needs a different IP address.
Subnet Mask	Enter the subnet mask of the network where the BDI is connected to. Use the following format: xxx.xxx.xxx.xxxe.g.255.255.255.0 A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI3000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This name is transmitted to the TFTP server when reading the configuration file.
Transmit	Click on this button to store the configuration in the BDI3000 flash memory.

Note:
Using this setup tool via the Network channel is only possible if the BDI3000 is already in Loader mode (Mode LED blinking). To force Loader mode, enter "boot loader" at the Telnet. The setup tool tries first to establish a connection to the Loader via the IP address present in the "BDI IP Address" entry field. If there is no connection established after a time-out, it tries to connect to the default IP (192.168.53.72).

© Copyright 1997-2015 by ABATRON AG Switzerland

2.5.3 Configuration via Telnet / TFTP

The firmware update and the initial configuration of the BDI3000 can also be done interactively via a Telnet connection and a running TFTP server on the host with the firmware file. In cases where it is not possible to connect to the default IP, the initial setup has to be done via a serial connection.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.

Following the steps to bring-up a new BDI3000 or updating the firmware.
Connect to the BDI Loader via Telnet.
If a firmware is already running enter "boot loader" and reconnect via Telnet.

```
$ telnet 192.168.53.72
or
$ telnet <your BDI IP address>
```

Update the network parameters so it matches your needs:

```
LDR>network
   BDI MAC     : 00-0c-01-30-00-01
   BDI IP      : 192.168.53.72
   BDI Subnet  : 255.255.255.0
   BDI Gateway : 255.255.255.255
   Config IP   : 255.255.255.255
   Config File :

LDR>netip 151.120.25.102
LDR>nethost 151.120.25.112
LDR>netfile /bdi3000/mytarget.cfg


LDR>network
   BDI MAC     : 00-0c-01-30-00-01
   BDI IP      : 151.120.25.102
   BDI Subnet  : 255.255.255.0
   BDI Gateway : 255.255.255.255
   Config IP   : 151.120.25.112
   Config File : /bdi3000/mytarget.cfg

LDR>network save
saving network configuration ... passed
   BDI MAC     : 00-0c-01-30-00-01
   BDI IP      : 151.120.25.102
   BDI Subnet  : 255.255.255.0
   BDI Gateway : 255.255.255.255
   Config IP   : 151.120.25.112
   Config File : /bdi3000/mytarget.cfg
```

In case the subnet has changed, reboot before trying to load the firmware

```
LDR>boot loader
```

© Copyright 1997-2015 by ABATRON AG Switzerland

Connect again via Telnet and program the firmware into the BDI flash:

```
$ telnet 151.120.25.102

LDR>info
   BDI Firmware: not loaded
   BDI CPLD ID : 01285043
   BDI CPLD UES: ffffffff
   BDI MAC     : 00-0c-01-30-00-01
   BDI IP      : 151.120.25.102
   BDI Subnet  : 255.255.255.0
   BDI Gateway : 255.255.255.255
   Config IP   : 151.120.25.112
   Config File : /bdi3000/mytarget.cfg

LDR>fwload e:/temp/b30pwsgd.100
erasing firmware flash ... passed
programming firmware flash ... passed

LDR>info
   BDI Firmware: 29 / 1.00
   BDI CPLD ID : 01285043
   BDI CPLD UES: ffffffff
   BDI MAC     : 00-0c-01-30-00-01
   BDI IP      : 151.120.25.102
   BDI Subnet  : 255.255.255.0
   BDI Gateway : 255.255.255.255
   Config IP   : 151.120.25.112
   Config File : /bdi3000/mytarget.cfg
LDR>
```

To boot now into the firmware use:

```
LDR>boot
```

The Mode LED should go off, and you can try to connect to the BDI again via Telnet.

```
telnet 151.120.25.102
```

© Copyright 1997-2015 by ABATRON AG Switzerland

2.6 Testing the BDI3000 to host connection

After the initial setup is done, you can test the communication between the host and the BDI3000. There is no need for a target configuration file and no TFTP server is needed on the host.

- If not already done, connect the BDI3000 system to the network.
- Power-up the BDI3000.
- Start a Telnet client on the host and connect to the BDI3000 (the IP address you entered during initial configuration).
- If everything is okay, a sign on message like «BDI Debugger for Embedded PowerPC» and a list of the available commands should be displayed in the Telnet window.

2.7 TFTP server for Windows

The bdiGDB system uses TFTP to access the configuration file and to load the application program. Because there is no TFTP server bundled with Windows, Abatron provides a TFTP server application **tftpsrv.exe**. This WIN32 console application runs as normal user application (not as a system service).

Command line syntax: tftpsrv [p] [w] [dRootDirectory]

Without any parameter, the server starts in read-only mode. This means, only read access request from the client are granted. This is the normal working mode. The bdiGDB system needs only read access to the configuration and program files.

The parameter [p] enables protocol output to the console window. Try it.
The parameter [w] enables write accesses to the host file system.
The parameter [d] allows to define a root directory.

tftpsrv p	Starts the TFTP server and enables protocol output
tftpsrv p w	Starts the TFTP server, enables protocol output and write accesses are allowed.
tftpsrv dC:\tftp\	Starts the TFTP server and allows only access to files in C:\tftp and its subdirectories. As file name, use relative names. For example "bdi\mpc7450.cfg" accesses "C:\tftp\bdi\mpc7450.cfg"

You may enter the TFTP server into the Startup group so the server is started every time you login.

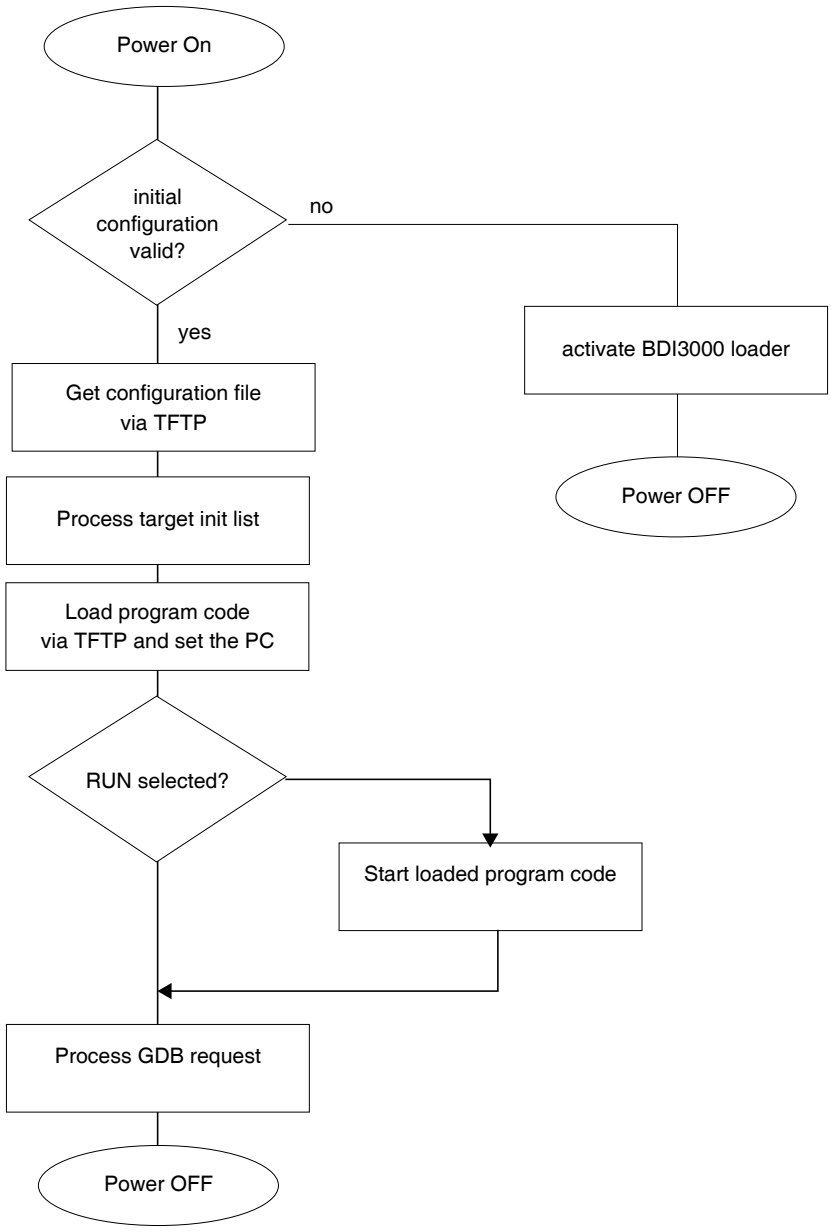
© Copyright 1997-2015 by ABATRON AG Switzerland

3 Using bdiGDB

3.1 Principle of operation

The firmware within the BDI handles the GDB request and accesses the target memory or registers via the JTAG interface. There is no need for any debug software on the target system. After loading the code via TFTP, debugging can begin at the very first assembler statement.

Whenever the BDI system is powered-up the following sequence starts:



© Copyright 1997-2015 by ABATRON AG Switzerland

Breakpoints:
There are two breakpoint modes supported. One of them (SOFT) is implemented by replacing application code with a TRAP instruction. The other (HARD) uses the built in breakpoint logic. If HARD is used, only 1 breakpoint can be active at the same time.
The following example selects SOFT as the breakpoint mode:

```
BREAKMODE   SOFT     ;SOFT or HARD, HARD uses PPC hardware breakpoints
```

All the time the application is suspended (i.e. caused by a breakpoint) the target processor remains freezed.

© Copyright 1997-2015 by ABATRON AG Switzerland

3.2 Configuration File

The configuration file is automatically read by the BDI after every power on.
The syntax of this file is as follows:

```
; comment
[part name]
identifier parameter1 parameter2 ..... parameterN  ; comment
identifier parameter1 parameter2 ..... parameterN
.....
[part name]
identifier parameter1 parameter2 ..... parameterN
identifier parameter1 parameter2 ..... parameterN
.....
                    etc.
```

Numeric parameters can be entered as decimal (e.g. 700) or as hexadecimal (0x80000).

Note about how to enter 64bit values:

The syntax for 64 bit parameters is : [<high word>_]<low word>

The "high word" (optional) and "low word" can be entered as decimal or hexadecimal. They are handled as two separate values concatenated with an underscore.

Examples:

0x01234567_0x89abcdef	=>>	0x0123456789abcdef
1_0	=>>	0x0000000100000000
256	=>>	0x0000000000000100
3_0x1234	=>>	0x0000000300001234
0x80000000_0	=>>	0x8000000000000000

© Copyright 1997-2015 by ABATRON AG Switzerland

3.2.1 Part [INIT]

The part [INIT] defines a list of commands which should be executed every time the target comes out of reset. The commands are used to get the target ready for loading the program file.

WGPR register value — Write value to the selected general purpose register.
- register — the register number 0 .. 31
- value — the value to write into the register

Example: WGPR 0 5

WSPR register value — Write value to the selected special purpose register.
- register — the register number
- value — the value to write into the register

Example: WSPR 27 0x00001002 ; SRR1 : ME,RI

WSR register value — Write value to the selected segment register.
- register — the register number
- value — the value to write into the register

Example: WSR 0 0x00001002 ; SR0 :

WREG name value — Write value to the selected CPU register by name
- name — the register name (MSR,CR,PC,XER,LR,CTR, ...)
- value — the value to write into the register

Example: WREG MSR 0x00001002

DELAY value — Delay for the selected time. A delay may be necessary to let the clock PLL lock again after a new clock rate is selected.
- value — the delay time in milliseconds (1...30000)

Example: DELAY 500 ; delay for 0.5 seconds

WM8 address value — Write a byte (8bit) to the selected memory place.
- address — the memory address
- value — the value to write to the target memory

Example: WM8 0xFFFFFA21 0x04 ; SYPCR: watchdog disable ...

WM16 address value — Write a half word (16bit) to the selected memory place.
- address — the memory address
- value — the value to write to the target memory

Example: WM16 0x02200200 0x0002 ; TBSCR

WM32 address value — Write a word (32bit) to the selected memory place.
- address — the memory address
- value — the value to write to the target memory

Example: WM32 0x02200000 0x01632440 ; SIUMCR

© Copyright 1997-2015 by ABATRON AG Switzerland

WM64 address value

Write a double word (64bit) to the selected memory place. This entry is mainly used to unlock flash blocks. The pattern written is generated by duplicating the value (0x12345678 -> 0x1234567812345678).

address the memory address
value the value used to generate the pattern
Example: WM64 0xFFF00000 0x00600060 ; unlock block 0

RM8 address value

Read a byte (8bit) from the selected memory place.

address the memory address
Example: RM8 0x00000000

RM16 address value

Read a half word (16bit) from the selected memory place.

address the memory address
Example: RM16 0x00000000

RM32 address value

Read a word (32bit) from the selected memory place.

address the memory address
Example: RM32 0x00000000

RM64 address value

Read a double word (64bit) from the selected memory place.

address the memory address
Example: RM64 0x00000000

TSZ1 start end

Defines a memory range with 1 byte maximal transfer size.
Normally when the BDI reads or writes a memory block, it tries to access the memory with a burst access. The TSZx entry allows to define a maximal transfer size for up to 8 address ranges.

start the start address of the memory range
end the end address of the memory range
Example: TSZ1 0xFF000000 0xFFFFFFFF ; PCI ROM space

TSZ2 start end

Defines a memory range with 2 byte maximal transfer size.

TSZ4 start end

Defines a memory range with 4 byte maximal transfer size.

TSZ8 start end

Defines a memory range with 8 byte maximal transfer size (no burst).

MMAP start end

Because a memory access to an invalid memory space via JTAG can lead to a deadlock, this entry can be used to define up to 32 valid memory ranges. If at least one memory range is defined, the BDI checks against this range(s) and avoids accessing of not mapped memory ranges.

start the start address of a valid memory range
end the end address of this memory range
Example: MMAP 0xFFE00000 0xFFFFFFFF ;Boot ROM

© Copyright 1997-2015 by ABATRON AG Switzerland

3.2.2 Part [TARGET]

The part [TARGET] defines some target specific values.
For MPC86xx, do not use parameters marked with a *.

CPUTYPE type [*NOBURSTREAD]

This value gives the BDI information about the connected CPU. If necessary, burst memory read accesses via JTAG can be disabled but this will slow down memory read performance dramatically.

type	7450,7451,7441,7455,7445,7457, 7447, 7448, 8641, 8610
Example:	CPUTYPE 7450

JTAGCLOCK value

With this value you select the JTAG clock frequency.

value	The JTAG clock frequency in Hertz or an index value from the following table: 0 = 32 MHz 2 = 11 MHz 4 = 5 MHz 1 = 16 MHz 3 = 8 MHz 5 = 4 MHz
Example:	JTAGCLOCK 1 ; JTAG clock is 16 MHz

BDIMODE mode [param]

This parameter selects the BDI debugging mode. The following modes are supported:

LOADONLY	Loads and starts the application core. No debugging via JTAG port.
AGENT	The debug agent runs within the BDI. There is no need for any debug software on the target. This mode accepts a second parameter. If RUN is entered as a second parameter, the loaded application will be started immediately, otherwise only the PC is set and BDI waits for GDB requests.
Example:	BDIMODE AGENT RUN

STARTUP mode [runtime][mode]

This parameter selects the target startup mode. The second mode defines how to handle the second e600 core in the MPC8641D. Not all mode combinations are supported. See chapter about Dual-Core support.
The following modes are supported:

HALT	This default mode forces the target to debug mode immediately out of reset. No code is executed after reset.
STOP	In this mode, the BDI lets the target execute code for "runtime" milliseconds after reset. This mode is useful when monitor code should initialize the target system.
RUN	After reset, the target executes code until stopped by the Telnet "halt" command.
Example:	STARTUP STOP 3000 ; let the CPU run for 3 seconds

BOOTADDR address

The boot address for PowerPC is 0xFFF00100. By default the BDI sets a hardware breakpoint at this address to freeze the processor immediately out of reset. You may change this initial breakpoint address.

address	the address where to set the startup breakpoint
Example:	BOOTADDR 0xFFF00120

© Copyright 1997-2015 by ABATRON AG Switzerland

WORKSPACE address

The BDI needs a workspace of 256 bytes in target RAM for code sequences to flush the data cache and to access L3 private memory. See also DCACHE and L3PM configuration parameter. If no workspace is defined, DCACHE is not flused and you cannot access L3 private memory.

address	the address of the RAM area
Example:	WORKSPACE 0x00000000

BREAKMODE mode [V]

This parameter defines how GDB requested breakpoints are implemented. The current mode can also be changed via the Telnet interface.
The V option forces the setting of the IABR[TE] / DABT[BT] bit.

SOFT	This is the normal mode. Breakpoints are implemented by replacing code with a TRAP instruction.
HARD	In this mode, the PPC breakpoint hardware is used. Only 1 breakpoint at a time is supported (IABR).
Example:	BREAKMODE HARD

STEPMODE mode

This parameter defines how single step (instruction step) is implemented. The alternate step mode (HWBP) may be useful when stepping instructions that causes a TLB miss exception.

TRACE	This is the default mode. Single step is implemented by setting the SE bit in MSR.
HWBP	In this mode, a hardware breakpoint on the next instruction is used to implement single stepping.
Example:	STEPMODE HWBP

VECTOR CATCH

When this line is present, the BDI catches all unhandled exceptions. Catching exceptions is only possible if the memory at address 0x00000000 to 0x00001FFF is writable.

Example:	VECTOR CATCH ; catch unhandled exception

*DCACHE mode

This parameter defines if the BDI flushes the caches before it accesses memory. If the BDI does not flush the caches, it executes L1/L2 cache coherent memory accesses. If the L1/L2 cache is enabled and the appropriate data is valid in the cache, data is read from the cache. For a write access, the cache is updated and the data also written to external memory. If there is an enabled L3 cache, flushing the data cache is recommended. Otherwise the debugger may display wrong data and working with software breakpoints may also fail. The following modes are supported:

NOFLUSH	The caches are not flushed. L1/L2 cache coherent memory accesses are used. Recommended if there is no enabled L3 cache in the system.
FLUSH	Before the BDI accesses any memory, the caches are flushed and only external memory is accessed. This mode needs a valid workspace for the flush code.
Example:	DCACHE NOFLUSH ; do not flush caches

© Copyright 1997-2015 by ABATRON AG Switzerland

POWERUP delay

When the BDI detects target power-up, HRESET is forced immediately. This way no code from a boot ROM is executed after power-up. The value entered in this configuration line is the delay time in milliseconds the BDI waits before it begins JTAG communication. This time should be longer than the on-board reset circuit asserts HRESET.

delay	the power-up start delay in milliseconds
Example:	POWERUP 5000 ;start delay after power-up

RESET type [time]

Normally the BDI toggles HRESET during the reset sequence. If reset type is NONE, the BDI does not assert HRESET at all. If reset type is KEEP then HRESET is asserted during the whole target power-up cycle to prevent the execution of any maybe not present boot code.This entry can also be used to change the default reset time.

type	NONE HARD (default) KEEP (keep HRESET asserted during target power-up)
time	The time in milliseconds the BDI assert the reset signal.
Example:	RESET NONE ; no reset during startup RESET HARD 1000 ; assert RESET for 1 second

WAKEUP time

This entry in the init list allows to define a delay time (in ms) the BDI inserts between releasing the COP-HRESET line and starting communicating with the target. This init list entry may be necessary if COP-HRESET is delayed on its way to the PowerPC reset pin.

time	the delay time in milliseconds
Example:	WAKEUP 3000 ; insert 3sec wake-up time

*MEMDELAY clocks

For slow memory it may be necessary to increase the number of clocks used to execute a memory access cycle. If for example you cannot access boot ROM content with the default configuration of your memory controller, define additional memory access clocks.

clocks	additional number of CPU clocks for a memory access
Example:	MEMDELAY 2000 ; additional memory access clocks

*L3PM base size

Defines the base address and size of the L3 cache private memory. Because L3 cache private memory cannot be accessed directly via JTAG, the BDI loads some support code into the workspace and uses it to access this memory range. Therefore a workspace is necessary to access this memory range.

Example: L3PM 0x01000000 0x100000 ; 1MB L3 private memory

MMU XLAT [kb]

In order to support Linux kernel debugging when MMU is on, the BDI translates effective (virtual) to physical addresses. This translation is done based on the current MMU configuration (BAT's and page tables). If this configuration line is present and address relocation active (MSR bits IR/DR), the BDI translates the addresses received from GDB before it accesses physical memory. The optional parameter defines the kernel virtual base address (default is 0xC0000000) and is used for default address translation. For more information see also chapter "Embedded Linux MMU Support". Addresses entered at the Telnet are never translated. Translation can be probed with the Telnet command PHYS.

kb	The kernel virtual base address (KERNELBASE)
Example:	MMU XLAT ;enable address translation

© Copyright 1997-2015 by ABATRON AG Switzerland

PTBASE addr

This parameter defines the physical memory address where the BDI looks for the virtual address of the array with the two page table pointers. For more information see also chapter "Embedded Linux MMU Support".

addr	Physical address of the memory used to store the virtual address of the array with the two page table pointers.
Example:	PTBASE 0xf0

REGLIST list

With GDB version 5.0, the number of registers read from the target has been increased. Additional registers like SR's, BAT's and SPR's are requested when you select a specific PowerPC variant with the "set processor" command (see GDB source file rs6000-tdep.c). In order to be compatible with older GDB versions and to optimize the time spent to read registers, this parameter can be used. You can define which register group is really read from the target. By default STD and FPR are read and transferred. This default is compatible with older GDB versions. The following names are use to select a register group:

STD	The standard (old) register block. The FPR registers are not read from the target but transferred. You can't disable this register group.
FPR	The floating point registers are read and transferred.
SR	The segment registers.
BAT	The IBAT and DBAT registers
SPR	The additional special purpose registers
AUX	currently not used
ALL	Include all register groups
Example:	REGLIST STD ; only standard registers REGLIST STD FPR SPR ; all except SR and BAT

SIO port [baudrate]

When this line is present, a TCP/IP channel is routed to the BDI's RS232 connector. The port parameter defines the TCP port used for this BDI to host communication. You may choose any port except 0 and the default Telnet port (23). On the host, open a Telnet session using this port. Now you should see the UART output in this Telnet session. You can use the normal Telnet connection to the BDI in parallel, they work completely independent. Also input to the UART is implemented.

port	The TCP/IP port used for the host communication.
baudrate	The BDI supports 2400 ... 115200 baud
Example:	SIO 7 9600 ;TCP port for virtual IO

QACK LOW

When this line is present, the BDI forces the QACK pin (pin 2) on the COP connector low. By default this pin is not driven by the BDI. Maybe useful for PPC750 and PPC7400 targets.

Example:	QACK LOW ; force QACK low via COP connector

© Copyright 1997-2015 by ABATRON AG Switzerland

Daisy chained JTAG devices:
The BDI can also handle systems with multiple devices connected to the JTAG scan chain. In order to put the other devices into BYPASS mode and to count for the additional bypass registers, the BDI needs some information about the scan chain layout. Enter the number (count) and total instruction register (irlen) length of the devices present before the PowerPC chip (Predecessor). Enter the appropriate information also for the devices following the PowerPC chip (Successor):

SCANPRED count irlen — This value gives the BDI information about JTAG devices present before the PowerPC chip in the JTAG scan chain.

count	The number of preceding devices
irlen	The sum of the length of all preceding instruction registers (IR).
Example:	SCANPRED 1 8 ; one device with an IR length of 8

SCANSUCC count irlen — This value gives the BDI information about JTAG devices present after the PowerPC chip in the JTAG scan chain.

count	The number of succeeding devices
irlen	The sum of the length of all succeeding instruction registers (IR).
Example:	SCANSUCC 2 12 ; two device with an IR length of 8+4

© Copyright 1997-2015 by ABATRON AG Switzerland

3.2.3 Part [HOST]

The part [HOST] defines some host specific values.

IP ipaddress

The IP address of the host.

ipaddress the IP address in the form xxx.xxx.xxx.xxx

Example: IP 151.120.25.100

FILE filename

The default name of the file that is loaded into RAM using the Telnet 'load' command. This name is used to access the file via TFTP. If the filename starts with a $, this $ is replace with the path of the configuration file name.

filename the filename including the full path or $ for relative path.

Example: FILE F:\gnu\demo\ppc\test.elf
FILE $test.elf

FORMAT format [offset]

The format of the image file and an optional load address offset. If the image is already stored in ROM on the target, select ROM as the format. The optional parameter "offset" is added to any load address read from the image file.

format SREC, BIN, AOUT, ELF, IMAGE* or ROM

Example: FORMAT ELF
FORMAT ELF 0x10000

LOAD mode

In Agent mode, this parameters defines if the code is loaded automatically after every reset.

mode AUTO, MANUAL

Example: LOAD MANUAL

START address

The address where to start the program file. If this value is not defined and the core is not in ROM, the address is taken from the image file. If this value is not defined and the core is already in ROM, the PC will not be set before starting the program file. This means, the program starts at the normal reset address (0xFFF00100).

address the address where to start the program file

Example: START 0x1000

* Special IMAGE load format:
The IMAGE format is a special version of the ELF format used to load a Linux boot image into target memory. When this format is selected, the BDI loads not only the loadable segment as defined in the Program Header, it also loads the rest of the file up to the Section Header Table. The relationship between load address and file offset will be maintained throughout this process. This way, the compressed Linux image and a optional RAM disk image will also be loaded.

© Copyright 1997-2015 by ABATRON AG Switzerland

DEBUGPORT port [RECONNECT]

The TCP port GDB uses to access the target. If the RECONNECT parameter is present, an open TCP/IP connection (Telnet/GDB) will be closed if there is a connect request from the same host (same IP address).

port the TCP port number (default = 2001)

Example: DEBUGPORT 2001

PROMPT string

This entry defines a new Telnet prompt. The current prompt can also be changed via the Telnet interface.

Example: PROMPT PPC_2

DUMP filename

The default file name used for the Telnet DUMP command.

filename the filename including the full path

Example: DUMP dump.bin

TELNET mode

By default the BDI sends echoes for the received characters and supports command history and line editing. If it should not send echoes and let the Telnet client in "line mode", add this entry to the configuration file.

mode ECHO (default), NOECHO or LINE

Example: TELNET NOECHO ; use old line mode

© Copyright 1997-2015 by ABATRON AG Switzerland

3.2.4 Part [FLASH]

The Telnet interface supports programming and erasing of flash memories. The bdiGDB system has to know which type of flash is used, how the chip(s) are connected to the CPU and which sectors to erase in case the ERASE command is entered without any parameter.

CHIPTYPE type

This parameter defines the type of flash used. It is used to select the correct programming algorithm.

format AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16, AT49, AT49X8, AT49X16, STRATAX8, STRATAX16, MIRROR, MIRRORX8, MIRRORX16, M58X32, AM29DX16, AM29DX32

Example: CHIPTYPE AM29F

CHIPSIZE size

The size of **one** flash chip in bytes (e.g. AM29F010 = 0x20000). This value is used to calculate the starting address of the current flash memory bank.

size the size of one flash chip in bytes

Example: CHIPSIZE 0x80000

BUSWIDTH width

Enter the width of the memory bus that leads to the flash chips. Do not enter the width of the flash chip itself. The parameter CHIPTYPE carries the information about the number of data lines connected to one flash chip. For example, enter 16 if you are using two AM29F010 to build a 16bit flash memory bank.

with the width of the flash memory bus in bits (8 | 16 | 32 | 64)

Example: BUSWIDTH 16

FILE filename

The default name of the file that is programmed into flash using the Telnet 'prog' command. This name is used to access the file via TFTP. If the filename starts with a $, this $ is replace with the path of the configuration file name. This name may be overridden interactively at the Telnet interface.

filename the filename including the full path or $ for relative path.

Example: FILE F:\gnu\ppc\bootrom.hex
FILE $bootrom.hex

FORMAT format [offset]

The format of the file and an optional address offset. The optional parameter "offset" is added to any load address read from the program file.
You get the best programming performance when using a binary format (BIN, AOUT, ELF or IMAGE).

format SREC, BIN, AOUT, ELF or IMAGE

Example: FORMAT BIN 0x10000

© Copyright 1997-2015 by ABATRON AG Switzerland

WORKSPACE address

If a workspace is defined, the BDI uses a faster programming algorithm that runs out of RAM on the target system. Otherwise, the algorithm is processed within the BDI. The workspace is used for a 1kByte data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose.

address	the address of the RAM area
Example:	WORKSPACE 0x00000000

ERASE addr [increment count] [mode [wait]]

The flash memory may be individually erased or unlocked via the Telnet interface. In order to make erasing of multiple flash sectors easier, you can enter an erase list. All entries in the erase list will be processed if you enter ERASE at the Telnet prompt without any parameter. This list is also used if you enter UNLOCK at the Telnet without any parameters. With the "increment" and "count" option you can erase multiple equal sized sectors with one entry in the erase list.

address	Address of the flash sector, block or chip to erase
increment	If present, the address offset to the next flash sector
count	If present, the number of equal sized sectors to erase
mode	BLOCK, CHIP, UNLOCK Without this optional parameter, the BDI executes a sector erase. If supported by the chip, you can also specify a block or chip erase. If UNLOCK is defined, this entry is also part of the unlock list. This unlock list is processed if the Telnet UNLOCK command is entered without any parameters.
wait	The wait time in ms is only used for the unlock mode. After starting the flash unlock, the BDI waits until it processes the next entry.
Example:	ERASE 0xff040000 ;erase sector 4 of flash ERASE 0xff060000 ;erase sector 6 of flash ERASE 0xff000000 CHIP ;erase whole chip(s) ERASE 0xff010000 UNLOCK 100 ;unlock, wait 100ms ERASE 0xff000000 0x10000 7 ; erase 7 sectors

Example for the ADS8260 flash memory:

```
[FLASH]
CHIPTYPE    I28BX8      ;Flash type
CHIPSIZE    0x200000    ;The size of one flash chip in bytes (e.g. AM29F010 = 0x20000)
BUSWIDTH    32          ;The width of the flash memory bus in bits (8 | 16 | 32 | 64)
WORKSPACE   0x04700000  ;workspace in dual port RAM
FILE        E:\gnu\demo\ads8260\bootrom.hex  ;The file to program
ERASE       0xFF900000  ;erase sector 4 of flash SIMM (LH28F016SCT)
ERASE       0xFF940000  ;erase sector 5 of flash SIMM
ERASE       0xFF980000  ;erase sector 6 of flash SIMM
ERASE       0xFF9c0000  ;erase sector 7 of flash SIMM
```

The above erase list maybe replaces with:

```
ERASE       0xFF900000  0x40000  4 ; erase sector 4 to 7 of flash SIMM
```

© Copyright 1997-2015 by ABATRON AG Switzerland

Supported Flash Memories:
There are currently 3 standard flash algorithm supported. The AMD, Intel and Atmel AT49 algorithm. Almost all currently available flash memories can be programmed with one of this algorithm. The flash type selects the appropriate algorithm and gives additional information about the used flash.

For 8bit only flash:	AM29F (MIRROR), I28BX8, AT49
For 8/16 bit flash in 8bit mode:	AM29BX8 (MIRRORX8), I28BX8 (STRATAX8), AT49X8
For 8/16 bit flash in 16bit mode:	AM29BX16 (MIRRORX16), I28BX16 (STRATAX16), AT49X16
For 16bit only flash:	AM29BX16, I28BX16, AT49X16
For 16/32 bit flash in 16bit mode:	AM29DX16
For 16/32 bit flash in 32bit mode:	AM29DX32
For 32bit only flash:	M58X32

Some newer Spansion MirrorBit flashes cannot be programmed with the MIRRORX16 algorithm because of the used unlock address offset. Use S29M32X16 for these flashes.

The AMD and AT49 algorithm are almost the same. The only difference is, that the AT49 algorithm does not check for the AMD status bit 5 (Exceeded Timing Limits).
Only the AMD and AT49 algorithm support chip erase. Block erase is only supported with the AT49 algorithm. If the algorithm does not support the selected mode, sector erase is performed. If the chip does not support the selected mode, erasing will fail. The erase command sequence is different only in the 6th write cycle. Depending on the selected mode, the following data is written in this cycle (see also flash data sheets): 0x10 for chip erase, 0x30 for sector erase, 0x50 for block erase.
To speed up programming of Intel Strata Flash and AMD MirrorBit Flash, an additional algorithm is implemented that makes use of the write buffer. This algorithm needs a workspace, otherwise the standard Intel/AMD algorithm is used.

The following table shows some examples:

Flash	x 8	x 16	x 32	Chipsize
Am29F010	AM29F	-	-	0x020000
Am29F800B	AM29BX8	AM29BX16	-	0x100000
Am29DL323C	AM29BX8	AM29BX16	-	0x400000
Am29PDL128G	-	AM29DX16	AM29DX32	0x01000000
Intel 28F032B3	I28BX8	-	-	0x400000
Intel 28F640J3A	STRATAX8	STRATAX16	-	0x800000
Intel 28F320C3	-	I28BX16	-	0x400000
AT49BV040	AT49	-	-	0x080000
AT49BV1614	AT49X8	AT49X16	-	0x200000
M58BW016BT	-	-	M58X32	0x200000
SST39VF160	-	AT49X16	-	0x200000
Am29LV320M	MIRRORX8	MIRRORX16	-	0x400000

© Copyright 1997-2015 by ABATRON AG Switzerland

Note:
Some Intel flash chips (e.g. 28F800C3, 28F160C3, 28F320C3) power-up with all blocks in locked state. In order to erase/program those flash chips, use the init list to unlock the appropriate blocks:

```
WM16    0xFFF00000      0x0060          unlock block 0
WM16    0xFFF00000      0x00D0
WM16    0xFFF10000      0x0060          unlock block 1
WM16    0xFFF10000      0x00D0
        ....
WM16    0xFFF00000      0xFFFF          select read mode
```

or use the Telnet "unlock" command:

```
UNLOCK [<addr> [<delay>]]
```

addr	This is the address of the sector (block) to unlock
delay	A delay time in milliseconds the BDI waits after sending the unlock command to the flash. For example, clearing all lock-bits of an Intel J3 Strata flash takes up to 0.7 seconds.

If "unlock" is used without any parameter, all sectors in the erase list with the UNLOCK option are processed.

To clear all lock-bits of an Intel J3 Strata flash use for example:

```
BDI> unlock 0xFF000000 1000
```

To erase or unlock multiple, continuous flash sectors (blocks) of the same size, the following Telnet commands can be used:

```
ERASE  <addr> <step> <count>
UNLOCK <addr> <step> <count>
```

addr	This is the address of the first sector to erase or unlock.
step	This value is added to the last used address in order to get to the next sector. In other words, this is the size of one sector in bytes.
count	The number of sectors to erase or unlock.

The following example unlocks all 256 sectors of an Intel Strata flash (28F256K3) that is mapped to 0x00000000. In case there are two flash chips to get a 32bit system, double the "step" parameter.

```
BDI> unlock 0x00000000 0x20000 256
```

© Copyright 1997-2015 by ABATRON AG Switzerland

3.2.5 Part [REGS]

In order to make it easier to access target registers via the Telnet interface, the BDI can read in a register definition file. In this file, the user defines a name for the register and how the BDI should access it (e.g. as memory mapped, memory mapped with offset, ...). The name of the register definition file and information for different registers type has to be defined in the configuration file. The register name, type, address/offset/number and size are defined in a separate register definition file.

An entry in the register definition file has the following syntax:

```
name    type    addr    [size [SWAP]]
```

name	The name of the register (max. 15 characters)
type	The register type GPR — General purpose register SPR — Special purpose register CCSR — Relative to CCSRBAR memory mapped register. The BDI knows the current position of the CCSR space. MM — Absolute direct memory mapped register DMM1...DMM4 — Relative direct memory mapped register IMM1...IMM4 — Indirect memory mapped register
addr	The address, offset or number of the register
size	The size (8, 16, 32) of the register (default is 32)
SWAP	If present, the bytes of a 16bit or 32bit register are swapped. This is useful to access little endian ordered registers (e.g. PCI bridge configuration registers).

The following entries are supported in the [REGS] part of the configuration file:

FILE filename	The name of the register definition file. This name is used to access the file via TFTP. The file is loaded once during BDI startup. filename — the filename including the full path Example: FILE C:\bdi\regs\mpc8260.def
DMMn base	This defines the base address of direct memory mapped registers. This base address is added to the individual offset of the register. base — the base address Example: DMM1 0x01000
IMMn addr data	This defines the addresses of the memory mapped address and data registers of indirect memory mapped registers. The address of a IMMn register is first written to "addr" and then the register value is access using "data" as address. addr — the address of the Address register data — the address of the Data register Example: DMM1 0x04700000

Remark:
The registers **msr**, **cr, iar** and **acc** and are predefined.

© Copyright 1997-2015 by ABATRON AG Switzerland

Example for a register definition (MPC107):

Entry in the configuration file:

```
[REGS]
DMM1      0xFC000000              ;Embedded utility memory base address
IMM1      0xFEC00000  0xFEE00000     ;configuration registers at byte offset 0
IMM2      0xFEC00000  0xFEE00001     ;configuration registers at byte offset 1
IMM3      0xFEC00000  0xFEE00002     ;configuration registers at byte offset 2
IMM4      0xFEC00000  0xFEE00003     ;configuration registers at byte offset 3
FILE      E:\cygnus\root\usr\demo\sp7450\mpc107.def
```

The register definition file:

```
;name                type       addr                 size
;-----------------------------------------------
;
gpr0                 GPR        0
sp                   GPR        1
;
xer                  SPR        1
lr                   SPR        8
ctr                  SPR        9
sprg0                SPR        272
sprg1                SPR        273
sprg2                SPR        274
sprg3                SPR        275
....
;
;
; IMMx must be set to the configuration registers
;
vendor               IMM1       0x00000080           16 SWAP
device               IMM3       0x00000080           16 SWAP
;
lmbar                IMM1       0x10000080           32 SWAP
;
pmcr1                IMM1       0x70000080           16 SWAP
pmcr2                IMM3       0x70000080           8
odcr                 IMM4       0x70000080           8
cdcr                 IMM1       0x74000080           16 SWAP
eumbbar              IMM1       0x78000080           32 SWAP
msar1                IMM1       0x80000080           32 SWAP
....
```

Now the defined registers can be accessed by name via the Telnet interface:

```
BDI> rd lmbar
BDI> rm sprg0 0xFF801801
```

© Copyright 1997-2015 by ABATRON AG Switzerland

3.3 Debugging with GDB

Because the target agent runs within BDI, no debug support has to be linked to your application. There is also no need for any BDI specific changes in the application sources. Your application must be fully linked because no dynamic loading is supported.

3.3.1 Target setup

Target initialization may be done at two places. First with the BDI configuration file, second within the application. The setup in the configuration file must at least enable access to the target memory where the application will be loaded. Disable the watchdog and setting the CPU clock rate should also be done with the BDI configuration file. Application specific initializations like setting the timer rate are best located in the application startup sequence.

3.3.2 Connecting to the target

As soon as the target comes out of reset, BDI initializes it and loads your application code. If RUN is selected, the application is immediately started, otherwise only the target PC is set. BDI now waits for GDB request from the debugger running on the host.

After starting the debugger, it must be connected to the remote target. This can be done with the following command at the GDB prompt:

```
(gdb)target remote bdi3000:2001
```

bdi3000	This stands for an IP address. The HOST file must have an appropriate entry. You may also use an IP address in the form xxx.xxx.xxx.xxx
2001	This is the TCP port used to communicate with the BDI

If not already suspended, this stops the execution of application code and the target CPU changes to background debug mode.
Remember, every time the application is suspended, the target CPU is freezed. During this time, no hardware interrupts will be processed.

Note: For convenience, the GDB detach command triggers a target reset sequence in the BDI.

```
(gdb)...
(gdb)detach
... Wait until BDI has resetet the target and reloaded the image
(gdb)target remote bdi3000:2001
```

Note:
After loading a program to the target you cannot use the GDB "*run*" command to start execution. You have to use the GDB "*continue*" command.

© Copyright 1997-2015 by ABATRON AG Switzerland

3.3.3 Breakpoint Handling

GDB versions before V5.0:
GDB inserts breakpoints by replacing code via simple memory read / write commands. There is no command like "Set Breakpoint" defined in the GDB remote protocol. When breakpoint mode HARD is selected, the BDI checks the memory write commands for such hidden "Set Breakpoint" actions. If such a write is detected, the write is not performed and the BDI sets an appropriate hardware breakpoint. The BDI assumes that this is a "Set Breakpoint" action when memory write length is 4 bytes and the pattern to write is 0x7D821008 (tw 12,r2,r2).

GDB version V5.x:
GDB version >= 5.0 uses the Z-packet to set breakpoints (watchpoints). For software breakpoints, the BDI replaces code with 0x7D821008 (tw 12,r2,r2). When breakpoint mode HARD is selected, the BDI sets an appropriate hardware breakpoint.

3.3.4 GDB monitor command

The BDI supports the GDB V5.x "monitor" command. Telnet commands are executed and the Telnet output is returned to GDB. This way you can for example switch the BDI breakpoint mode from within your GDB session.

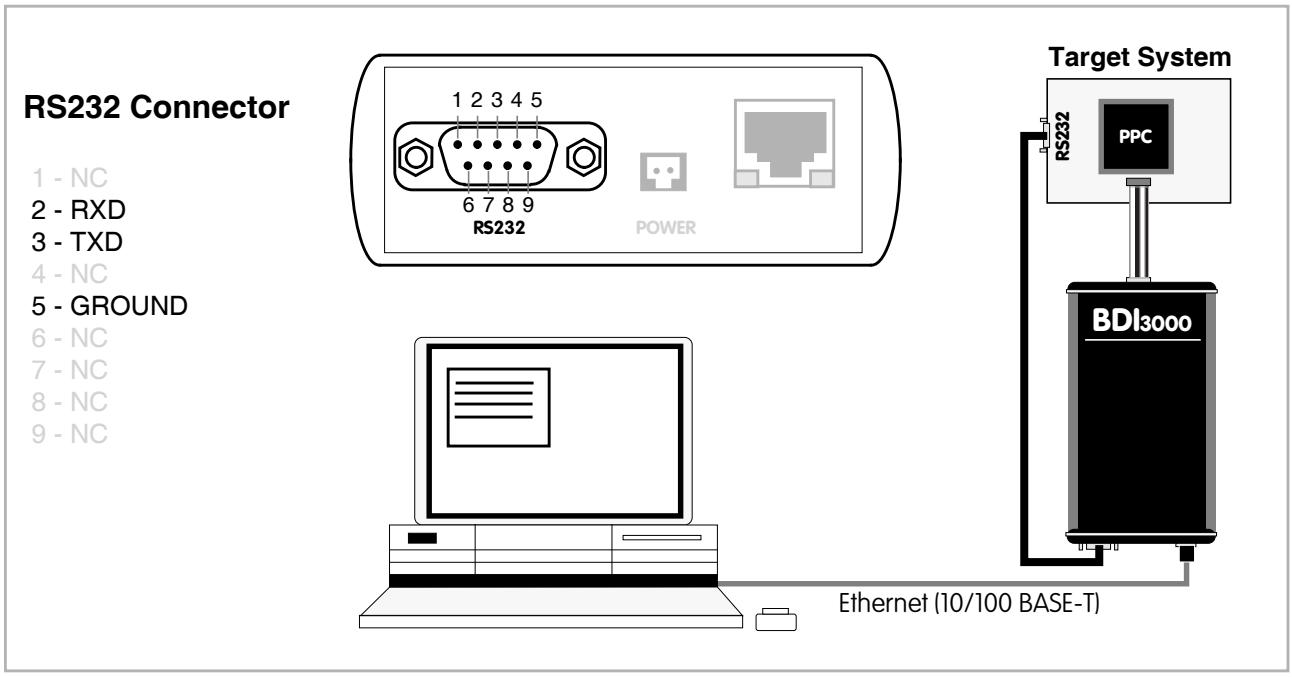
```
(gdb) target remote bdi3000:2001
Remote debugging using bdi3000:2001
0x10b2 in start ()
(gdb) monitor break
Breakpoint mode is SOFT
(gdb) mon break hard

(gdb) mon break
Breakpoint mode is HARD
(gdb)
```

© Copyright 1997-2015 by ABATRON AG Switzerland

3.3.5 Target serial I/O via BDI

A RS232 port of the target can be connected to the RS232 port of the BDI3000. This way it is possible to access the target's serial I/O via a TCP/IP channel. For example, you can connect a Telnet session to the appropriate BDI3000 port. Connecting GDB to a GDB server (stub) running on the target should also be possible.



The configuration parameter "SIO" is used to enable this serial I/O routing.
The used framing parameters are 8 data, 1 stop and not parity.

```
[TARGET]
....
SIO      7        9600              ;Enable SIO via TCP port 7 at 9600 baud
```

Warning!!!
Once SIO is enabled, connecting with the setup tool to update the firmware will fail. In this case either disable SIO first or disconnect the BDI from the LAN while updating the firmware.

© Copyright 1997-2015 by ABATRON AG Switzerland

3.3.6 Embedded Linux MMU Support

The bdiGDB system supports Linux kernel debugging when MMU is on. The MMU configuration parameter enables this mode of operation. In this mode, all addresses received from GDB are assumed to be virtual. Before the BDI accesses memory, it translates this address into a physical one based on information found in the kernel/user page table.

In order to search the page tables, the BDI needs to know the start addresses of the first level page table. The configuration parameter PTBASE defines the physical address where the BDI looks for the virtual address of an array with two virtual addresses of first level page tables. The first one points normally to the kernel page table, the second one can point to the current user page table. As long as the base pointer or the first entry is zero, the BDI does only BAT and default translation. Default translation maps a 256 Mbyte range starting at KERNELBASE to 0x00000000. The second page table is only searched if its address is not zero and there was no match in the first one.

The pointer stucture is as follows:

```
PTBASE (physical address) ->
        PTE pointer pointer(virtual or physical address) ->
                PTE kernel pointer (virtual or physical address)
                PTE user pointer (virtual or physical address)
```

Newer versions of "arch/ppc/kernel/head.S" support the automatic update of the BDI page table information structure. Search "head.S" for "abatron" and you will find the BDI specific extensions.

Extract from the configuration file:

```
[INIT]
......
WM32    0x000000f0      0x00000000      ;invalidate page table base

[TARGET]
....
MMU         XLAT        ;translate effective to physical address
PTBASE      0x000000f0  ;here is the pointer to the page table pointers
```

© Copyright 1997-2015 by ABATRON AG Switzerland

To debug the Linux kernel when MMU is enabled you may use the following load and startup sequence:

- Load the compressed linux image
- Set a hardware breakpoint with the Telnet at a point where MMU is enabled. For example at "start_kernel".
  BDI> BI 0xC0061550 v
- Start the code with GO at the Telnet
- The Linux kernel is decompressed and started
- The system should stop at the hardware breakpoint (e.g. at start_kernel)
- Disable the hardware breakpoint with the Telnet command CI.
- If not automatically done by the kernel, setup the page table pointers for the BDI.
- Start GDB with vmlinux as parameter
- Attach to the target
- Now you should be able to debug the Linux kernel

To setup the BDI page table information structure manually, set a hardware breakpoint at "start_kernel" and use the Telnet to write the address of "swapper_pg_dir" to the appropriate place.

```
BDI>bi 0xc0061550              /* set breakpoint at start_kernel */
BDI>go
..                             /* target stops at start_kernel */
BDI>ci
BDI>mm 0xf0 0xc00000f8         /* Let PTBASE point to an array of two pointers*/
BDI>mm 0xf8 0xc0057000         /* write address of swapper_pg_dir to first pointer */
BDI>mm 0xfc 0x00000000         /* clear second (user) pointer */
```

© Copyright 1997-2015 by ABATRON AG Switzerland

3.4 Telnet Interface

A Telnet server is integrated within the BDI. The Telnet channel is used by the BDI to output error messages and other information. Also some basic debug commands can be executed.

Telnet Debug features:

- Display and modify memory locations
- Display and modify general and special purpose registers
- Single step a code sequence
- Set hardware breakpoints
- Load a code file from any host
- Start / Stop program execution
- Programming and Erasing Flash memory

During debugging with GDB, the Telnet is mainly used to reboot the target (generate a hardware reset and reload the application code). It may be also useful during the first installation of the bdiGDB system or in case of special debug needs.

Multiple commands separated by a semicolon can be entered on one line.

Example of a Telnet session:

```
BDI>res
- TARGET: processing user reset request
- TARGET: Target PVR is 0x80000200
- TARGET: reseting target passed
- TARGET: processing target init list ....
- TARGET: processing target init list passed
BDI>info
   Target CPU       : MPC7450 Rev.2
   Target state     : debug mode
   Debug entry cause : COP freeze (startup)
   Current PC       : 0xfff00100
   Current CR       : 0x00000000
   Current MSR      : 0x00000000
   Current LR       : 0x00000000
BDI>md 0xfff00100
fff00100 : 48003270 60000000 60000000 60000000  H.. `...`...`...
fff00110 : 60000000 60000000 60000000 60000000  `...`...`...`...
                        ......
```

Notes:
The DUMP command uses TFTP to write a binary image to a host file. Writing via TFTP on a Linux/ Unix system is only possible if the file already exists and has public write access. Use "man tftpd" to get more information about the TFTP server on your host.

The BI command sets a hardware breakpoint via the IABR register. IABR[TE] must be equal MSR[IR] in order for a match to be signalled. IABR[TE] is set when the parameter V is present in the BREAKMODE configuration. Otherwise it is cleared. You can override this default setting with the optional parameter v (virtual, sets TE) or p (physical, clears TE).

© Copyright 1997-2015 by ABATRON AG Switzerland

The Telnet commands:

```
"PHYS  <address>              converts an effective to a physical address",
"MD    [<address>] [<count>]  display target memory as word (32bit)",
"MDD   [<address>] [<count>]  display target memory as double word (64bit)",
"MDH   [<address>] [<count>]  display target memory as half word (16bit)",
"MDB   [<address>] [<count>]  display target memory as byte (8bit)",
"DUMP  <addr> <size> [<file>] dump target memory to a file",
"MM    <addr> <value> [<cnt>] modify word(s) (32bit) in target memory",
"MMD   <addr> <value> [<cnt>] modify double word(s) (64bit) in target memory",
"MMH   <addr> <value> [<cnt>] modify half word(s) (16bit) in target memory",
"MMB   <addr> <value> [<cnt>] modify byte(s) (8bit) in target memory",
"MC    [<address>] [<count>]  calculates a checksum over a memory range",
"MV                           verifies the last calculated checksum",
"RD    [<name>]               display general purpose or user defined register",
"RDUMP [<file>]               dump all user defined register to a file",
"RDFPR                        display floating point registers",
"RDSPR <number>               display special purpose register",
"RDSR  <number>               display segment register",
"RDVR  [<number>]             display vector register",
"RM    {<nbr>|<name>} <value> modify general purpose or user defined register",
"RMSPR <number>   <value>     modify special purpose register",
"RMSR  <number>   <value>     modify segment register",
"RMVR  <nbr><val val val val> modify vector register (four 32bit values)",
"ICACHE <addr | set>          display L1 instruction cache content",
"DCACHE <addr | set>          display L1 data cache content",
"L2CACHE <addr | set>         display L2 cache content",
"DTAG  <from> [<to>]          display L1 DTAG values",
"ITAG  <from> [<to>]          display L1 ITAG values",
"RESET [HALT | RUN [time]]    reset the target system, change startup mode",
"BREAK [SOFT | HARD]          display or set current breakpoint mode",
"GO    [<pc>]                 set PC and start target system",
"TI    [<pc>]                 trace on instuction (single step)",
"TC    [<pc>]                 trace on change of flow",
"HALT                         force target to enter debug mode",
"BI  <addr> [v|p]             set instruction hardware breakpoint",
"CI  [<id>]                   clear instruction hardware breakpoint(s)",
"BD  [R|W] <addr>             set data watchpoint via DABR (DABR[BT]=0)",
"BDT [R|W] <addr>             set data watchpoint via DABR (DABR[BT]=1)",
"CD  [<id>]                   clear data watchpoint(s)",
"INFO                         display information about the current state",
"LOAD   [<offset>] [<file> [<format>]] load program file to target memory",
"VERIFY [<offset>] [<file> [<format>]] verify a program file to target memory",
"PROG   [<offset>] [<file> [<format>]] program flash memory",
"                             <format> : SREC, BIN, AOUT or ELF",
"ERASE  [<address> [<mode>]]  erase a flash memory sector, chip or block",
"               <mode>  : CHIP, BLOCK or SECTOR (default is sector)",
"ERASE  <addr> <step> <count> erase multiple flash sectors",
"UNLOCK [<addr> [<delay>]]    unlock a flash sector",
"UNLOCK <addr> <step> <count> unlock multiple flash sectors",
"FLASH  <type> <size> <bus>   change flash configuration",
"DELAY  <ms>                  delay for a number of milliseconds",
"SELECT <core>                change the current core",
"HOST   <ip>                  change IP address of the host",
"PROMPT <string>              defines a new prompt string",
"CONFIG                       display or update BDI configuration",
"CONFIG <file> [<hostIP> [<bdiIP> [<gateway> [<mask>]]]]",
"HELP                         display command list",
"SAP    [0 | 1]               enable / disable JTAG memory access via SAP",
"BOOT                         reboot the BDI and reload the configuration",
"QUIT                         terminate the Telnet session"
```

© Copyright 1997-2015 by ABATRON AG Switzerland

3.5 Dual-Core Support for MPC8641D

The bdiGDB system supports concurrent debugging of the two e600 cores present in the MPC8641D. For every core you can start its own GDB session. The port numbers used to attach the remote targets are 2001 and 2002. In the Telnet you switch between the cores with the command "select {0 | 1}".

In the configuration file, simply begin the init list line with the appropriate core number. If there is no #n in front of a line, the BDI assumes core #0.

```
[INIT]
; init core register
#0 WREG    MSR          0x00001002     ;MSR : ME,RI
#0 WSPR    1008         0x84000000     ;HID0: disable cache, set TBEN bit
#0 WSPR    1017         0x00000000     ;L2CR: disable L2 cache
;
#1 WREG    MSR          0x00001002     ;MSR : ME,RI
#1 WSPR    1008         0x84000000     ;HID0: disable cache, set TBEN bit
```

The BDI supports different startup modes. The startup mode is defined via an entry in the [TARGET] section of the BDI configuration file. The second e600 core (core #1) is handled by the BDI only if there is a second "mode" parameter present in the STARTUP line. Because after reset the second core is disabled, the BDI writes to the MCMPCR and enables it in cases where HALT is selected as startup mode for the second core. Following some examples:

STARTUP HALT HALT
The second core will be enabled via MCMPCR and both core are halted at the reset vector via an IABR breakpoint.

STARTUP RUN RUN
Both core are let running after reset. You can halt them individually via the Telnet "halt" command. The BDI does not write to MCMPCR. Halting the second core will only succeed if it has been enabled form the code running on the first core. The init list is not processed in this case.

STARTUP STOP 4000 HALT
The second core will be enabled via MCMPCR and halted at the reset vector via an IABR breakpoint. The first core is let running for 4 seconds and the halted.

STARTUP STOP 4000 STOP
Both core are let running after reset for 4 seconds and the halted. The BDI does not write to MCMP-CR. Halting the second core will only succeed if it has been enabled form the code running on the first core during this 4 second runtime. After halting, the init list is processed.

STARTUP HALT RUN
Useful if you want to debug boot code on core#0 but want to be able to access core#1 later. The BDI does not write to MCMPCR in this case.

STARTUP RUN HALT
The first core is let running while the second core will be enabled via MCMPCR and halted at the reset vector via an IABR breakpoint.

© Copyright 1997-2015 by ABATRON AG Switzerland

Note about memory accesses via JTAG:
On MPC86xx targets, memory accesses are done via the so called "System Access Port" (SAP). The SAP is like an additional bus master. You can access memory while the core(s) are running and memory coherency is maintained because SAP accesses are snooped. This has the side effect that for example cache lines are flushed when memory is accessed via the BDI. Debugging code where data/stack is only present in the cache without real memory behind it becomes almost impossible.

The following Telnet sequence shows the effect on reading cached data via the BDI.

```
8641>dcache 0
W0 : 0_0010f000 VD 0010f000 0010f004 0010f008 0010f00c
             0010f010 0010f014 0010f018 0010f01c
W1 : 0_0010b000 VD 0010b000 0010b004 0010b008 0010b00c
             0010b010 0010b014 0010b018 0010b01c
W2 : 0_00111000 VD 00111000 00111004 00111008 0011100c
             00111010 00111014 00111018 0011101c
W3 : 0_0010d000 VD 0010d000 0010d004 0010d008 0010d00c
             0010d010 0010d014 0010d018 0010d01c

8641>md 0x00111000
0_00111000 : 00111000 00111004 00111008 0011100c  ................
0_00111010 : 00111010 00111014 00111018 0011101c  ................
.......
0_001110e0 : 001110e0 001110e4 001110e8 001110ec  ................
0_001110f0 : 001110f0 001110f4 001110f8 001110fc  ................

8641>dcache 0
W0 : 0_0010f000 VD 0010f000 0010f004 0010f008 0010f00c
             0010f010 0010f014 0010f018 0010f01c
W1 : 0_0010b000 VD 0010b000 0010b004 0010b008 0010b00c
             0010b010 0010b014 0010b018 0010b01c
W2 : 0_00111000 V- 00111000 00111004 00111008 0011100c
             00111010 00111014 00111018 0011101c
W3 : 0_0010d000 VD 0010d000 0010d004 0010d008 0010d00c
             0010d010 0010d014 0010d018 0010d01c
```

© Copyright 1997-2015 by ABATRON AG Switzerland

4 Specifications

Operating Voltage Limiting	5 VDC ± 0.25 V
Power Supply Current	typ. 500 mA max. 1000 mA
RS232 Interface: Baud Rates	9’600,19’200, 38’400, 57’600,115’200
Data Bits	8
Parity Bits	none
Stop Bits	1
Network Interface	10/100 BASE-T
BDM/JTAG clock	up to 32 MHz
Supported target voltage	1.2 – 5.0 V
Operating Temperature	+ 5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (noncondensing)	<90 %rF
Size	160 x 85 x 35 mm
Weight (without cables)	280 g
Host Cable length (RS232)	2.5 m
Electromagnetic Compatibility	CE compliant
Restriction of Hazardous Substances	RoHS 2002/95/EC compliant

Specifications subject to change without notice

© Copyright 1997-2015 by ABATRON AG Switzerland

5 Environmental notice

Disposal of the equipment must be carried out at a designated disposal site.

6 Declaration of Conformity (CE)

CE

DECLARATION OF CONFORMITY

This declaration is valid for following product:

Type of device: BDM/JTAG Interface
Product name: BDI3000

The signing authorities state, that the above mentioned equipment meets the requirements for emission and immunity according to

EMC Directive 89/336/EEC

The evaluation procedure of conformity was assured according to the following standards:

IEC 61000-6-2: 1999, mod. EN61000-6-2: 2001
IEC 61000-6-3: 1996, mod. EN61000-6-2: 2001

This declaration of conformity is based on the test report no. E1087-05-7a of Quinel, Zug, Swiss Testing Service, accreditation no. STS 037

Manufacturer:

ABATRON AG
Lettenstrasse 9
CH-6343 Rotkreuz

Authority:

Max Vock
Marketing Director

Ruedi Dummermuth
Technical Director

Rotkreuz, 7/18/2007

© Copyright 1997-2015 by ABATRON AG Switzerland

7 Warranty and Support Terms

7.1 Hardware

ABATRON Switzerland warrants the Hardware to be free of defects in materials and workmanship for a period of 3 years following the date of purchase when used under normal conditions. In the event of notification within the warranty period of defects in material or workmanship, ABATRON will repair or replace the defective hardware. The cost for the shipment to Abatron must be paid by the customer. Failure in handling which leads to defects are not covered under this warranty. The warranty is void under any self-made repair operation.

7.2 Software

License

Against payment of a license fee the client receives a usage license for this software product, which is not exclusive and cannot be transferred.

Copies

The client is entitled to make copies according to the number of licenses purchased. Copies exceeding this number are allowed for storage purposes as a replacement for defective storage mediums.

Update and Support

The agreement includes free software maintenance (update and support) for one year from date of purchase. After this period the client may purchase software maintenance for an additional year.

7.3 Warranty and Disclaimer

ABATRON AND ITS SUPPLIERS HEREBY DISCLAIMS AND EXCLUDES, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT.

7.4 Limitation of Liability

IN NO EVENT SHALL ABATRON OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING, WITHOUT LIMITATION, ANY SPECIAL, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE HARDWARE AND/OR SOFTWARE, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, BUSINESS, DATA, GOODWILL, OR ANTICIPATED SAVINGS, EVEN IF ADVISED OF THE POSSIBILITY OF THOSE DAMAGES.

The hardware and software product with all its parts, copyrights and any other rights remain in possession of ABATRON. Any dispute, which may arise in connection with the present agreement shall be submitted to Swiss Law in the Court of Zug to which both parties hereby assign competence.

© Copyright 1997-2015 by ABATRON AG Switzerland

Appendices

A Troubleshooting

Problem
The firmware can not be loaded.

Possible reasons

- The BDI is not correctly connected with the Host (see chapter 2).
- A wrong communication port is selected (Com 1...Com 4).
- The BDI is not powered up

Problem
No working with the target system (loading firmware is okay).

Possible reasons

- Wrong pin assignment (BDM/JTAG connector) of the target system (see chapter 2).
- Target system initialization is not correctly –> enter an appropriate target initialization list.
- An incorrect IP address was entered (BDI3000 configuration)
- BDM/JTAG signals from the target system are not correctly (short-circuit, break, ...).
- The target system is damaged.

Problem
Network processes do not function (loading the firmware was successful)

Possible reasons

- The BDI3000 is not connected or not correctly connected to the network (LAN cable or media converter)
- An incorrect IP address was entered (BDI3000 configuration)

© Copyright 1997-2015 by ABATRON AG Switzerland

B Maintenance

The BDI needs no special maintenance. Clean the housing with a mild detergent only. Solvents such as gasoline may damage it.

C Trademarks

All trademarks are property of their respective holders.

© Copyright 1997-2015 by ABATRON AG Switzerland